Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

DEVELOPMENT OF AN IR SYSTEM FOR ARGUMENT RETRIEVAL TO ANSWER CONTROVERSIAL QUESTIONS

Master Degree in Computer Engineering at University of Padua, Search Engines Course, Academic Year: 2021/2022

Group: LGTM, General Grievous

Barusco Manuel, Forzan Riccardo, Rizzetto Nicola, Soleymani Elham, Del Fiume Gabriele, Peloso Mario Giovanni

INTRODUCTION: TASK 1

- **Purpose**: support users who search for arguments to be used in conversations and debates on controversial topics

- **Task**: retrieve couples of strong "pro" or "con" sentences on the provided topic

- **Data**: args.me corpus, a collection of about 650.000 preprocessed documents extracted from web debate portals and merged in a single csv file

OUR APPROACH: PIPELINE & TECHNIQUES

Parsing/Pre-Processing/Indexing

- Parsing of the CSV file.
- Filtering of the main information extracted.
- Indexing of the parsed data.

Searching

- Searching with different retrieval models:
  - LMDirichlet
  - BM25

Augmentation

- Query Boosting.
- Query Expansion based on the most important query tokens.
- Re-ranking based on:
  - Sentiment Analysis
  - Readability

of the retrieved documents.

PARSING/ PRE-PROCESSING / INDEXING

Parsing: parsing of the CSV file structure and extraction of the most important informations fields

Pre-processing: application of different stoplists, filters and stemmers

Indexing: indexing of the pre-processing phase output tokens

PARSING:

Two libraries are used:

- **Jackson Library**
- **Apache CSV**

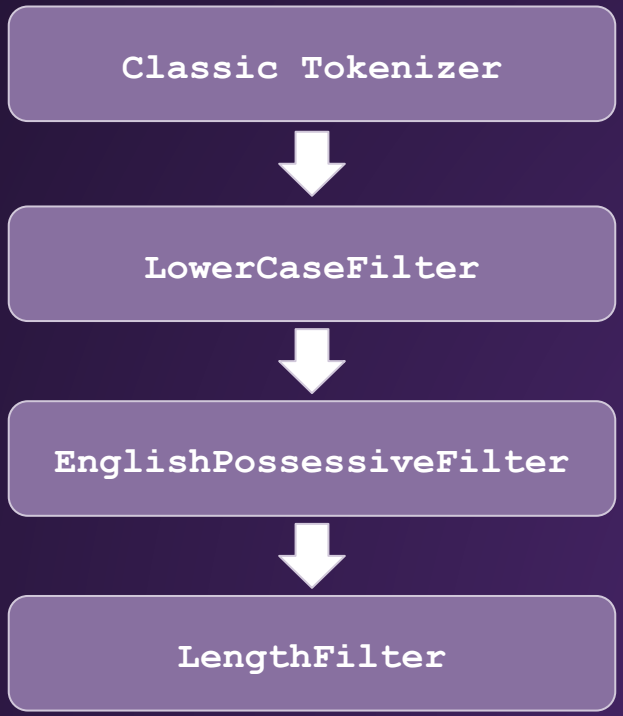
The most important informations fields extracted:

- ID, Conclusion and Stance of the Documents
- Discussion Title and Source Title of the Documents
- Text of the Documents
- Sentences ID

INDEXING: CUSTOM FIELDS

`IDKeyField`	For key informations such that ID's and Stance
`BodyCorrelatedField`	For document correlated informations such that DiscussionTitle and SourceTitle
`BodyField`	For the document text

PRE-PROCESSING: FILTERS



PREPROCESSING: DIFFERENT STOPLISTS

- Tried different stoplist
- Best scores with:
  - No Stop
  - CoreNLP

We decided not to use Stop Lists

StopList	nDCG@5	P_5	RECALL
CoreNLP	0.4240	0.3519	0.8507
CountWords Free	0.4240	0.3482	0.8496
EBSCO	0.4200	0.3522	0.8496
GoogleStop	0.4240	0.3519	0.8500
Ranks	0.4080	0.3362	0.8499
NO STOP	0.4200	0.3532	0.8524

PREPROCESSING: DIFFERENT STEMMERS

- Tried different stemmers on top of No Stop Lists solution
- Best score with:
  - No Stem

We decided not to use Stemmers

Stemmer	nDCG@5	P_5	RECALL
EnglishMinimal Stemmer	0.3532	0.4200	0.8534
KStemFilter	0.3833	0.4300	0.8600
PorterStemmer	0.3533	0.4243	0.8600
NO STEM	0.4200	0.3532	0.8600

SEARCHING: DIFFERENT RETRIEVAL MODELS

We tested two different similarities (on the final version of the system):

- BM25
- LMDirichlet μ= 1700
- LMDirichlet μ= 1800

Similarity	nDCG@5	P_5	RECALL
BM25	0.4428	0.5120	0.8436
LMDirichlet μ= 1700	0.3926	0.4720	0.8666
LMDirichlet μ= 1700	0.3870	0.4640	0.8663

We decided to use the BM25 Similarity

AUGMENTATION: QUERY BOOSTING

We search on 4 document fields (SourceTitle, DiscussionTitle, SourceText and Conclusion) so we tried to boost the search on the different fields by using different weights.

Source Text	Conclusion	Discussion Title	Source Title	nDCG@5	P_5	RECALL
0,25	1	1	1	0.2974	0.3760	0.7473
1	0,75	0.5	0.5	0.3225	0.3920	0.7473
1	1	1	0.5	0.2974	0.3760	0.7492
2	1	1	1	0.3455	0.4280	0.7481
4	1	1	1	0.3410	0.4240	0.7391
2	2	1	1	0.3257	0.3920	0.7473
2	0.5	1	1	0.3457	0.4280	0.7410

AUGMENTATION: QUERY EXPANSION

Find keywords

The RAKE algorithm was used to recognize the most significant tokens of a query

Find synonyms

DataMuse API are used to find synonyms for the extracted keywords

Generate new queries

Synonyms are used to generate other related queries

AUGMENTATION: QUERY EXPANSION

Expansion algorithm

- Each synonym retrieved by DataMuse API has a similarity score.
- Our query expansion algorithm is parametric so we can adjust the quality of the generated queries.

Expansion results

- The output queries are mixed with the source one.
- Duplicates are removed from the result list.
- The output set of queries is ordered by score.

AUGMENTATION: RE-RANKING

Re-Ranking

Re-ranking algorithm takes into account:

- Sentiment of the query and the conclusion field of documents retrieved
- Readability score of the documents retrieved

Details

We used an implementation of Valence Aware Dictionary and sEntiment Reasoner algorithm for judging the sentiment of text fields.

CONCLUSIONS

- Query expansion marginally improved our system (marginally better recall but loss in precision)
- Sentiment analysis has not produced better results: text fields were too short and it was inapplicable it on the corpus of the document (running time limitations)

System Configuration	nDCG@5	P_5	RECALL
Search with BOOST, NO QE, NO RE-R	0.4428	0.5120	0.8436
Search with BOOST, QE in all token, NO RE-R	0.3961	0.4640	0.7527
Search with BOOST, QE only main token, NO RE-R	0.4294	0.5000	0.7925
Search with BOOST, QE only main token, RE-R Sent	0.2383	0.2760	0.6304
Search with BOOST, QE only main token, RE-R Read	0.1416	0.1800	0.4684

FUTURE WORK

- Better Query Expansion:
  - NLP tools
  - Relevance or Pseudo Relevance Feedback

- Better Re-Rank:
  - Learning To Rank methodologies

THANKS FOR YOUR ATTENTION